UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/042,079	01/07/2002	Scott J. Broussard	AUS920010996US1	6556

35525 7590 08/26/2005

IBM CORP (YA)
C/O YEE & ASSOCIATES PC
P.O. BOX 802333
DALLAS, TX 75380

EXAMINER	
ROMANO, JOHN J	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 08/26/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/042,079	BROUSSARD, SCOTT J.	
	Examiner	Art Unit	
	John J. Romano	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *6/14/2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) *1-39* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Remarks

1. Applicant's amendment and response received June 14th, 2005, responding to the February 24th, 2005, Office action provided in the rejections of claims 1-39, wherein claims 1, 7, 8, 12, 14, 20, 21, 25, 27, 33, 34 and 38 have been amended. Claims 1-39 remain pending in the application and which have been fully considered by the examiner.

2. The examiner withdraws the objection to claim 26 in view of the Applicants' amendments.

3. In reply to the telephone interview summary, Applicant states that Examiner agreed that *Krishna* fails to teach or suggest the features of identifying all public classes included in the first software object-oriented software package, for each of said public classes, identifying all public entities included in each of said public classes, or removing all references to software that is defined in a second object-oriented software package from said public entities included in each of said public classes, as alleged in the Office Action, which Examiner respectfully disagrees. Examiner recalls advising Applicant to issue the arguments formally and stating that the examiner will consider them entirely and update the search if necessary. Moreover, in regard to this limitation, the office action cites *Green* as the prior art, not *Krishna* as described hereinbelow, and the previous office action.

Applicant arguing for the claims being patentable over Krishna in view of Green (see pages 13-26 of the amendment and response), and arguments pertaining to the

dependent claims are not persuasive, as will be addressed under Prior Art's Arguments – Rejections section at item 2 and the claim rejections below. Accordingly, Applicants' amendment necessitated additional clarifications, in light of the rejection of the claims over prior art provided in the previous Office action, to further point out that the prior art also discloses as such claimed limitations as now amended which will be provided and/or addressed under the item 2 below. Thus, the rejection of the claims over prior art in the previous Office action is maintained in light of the necessitated additional clarifications provided hereon and **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Prior Art's Arguments – Rejections

4. Applicant's arguments filed June 14th, 2005, in particular on pages 15-23, have been fully considered but they are not persuasive. For example,

(1) In regard to the argument that the Office Action alleges that *Krishna* teaches the features of identifying all public classes included in a first object-oriented software package and identifying all public entities included in each of the public classes, (page 13 of 23, second paragraph of the amendment and response), the examiner respectfully disagrees. The examiner would like to point out that *Green* is cited for teaching the above limitation as indicated in the previous office action and Applicants' response to amendment (Page 12 of 23, First Paragraph) as follows:

> But **Krishna** does not expressly disclose "*...identifying all public classes included in said first software package ...*" or "*...for each of said public classes, identifying all public entities included in each of said public classes ...*". However, **Green** discloses:
>
> - "*...identifying all public classes included in said first software package ...*" (E.g., see "Discovering Class Modifiers", Page 7), wherein all public class modifiers are discovered.
> - "*...for each of said public classes, identifying all public entities included in each of said public classes ...*" (E.g., see "Trail: The Reflection API", Page 1, Paragraph 1), wherein all public class modifiers are discovered along with their fields, methods and variables (entities).

Similarly, in regard to this argument as presented (Page 14 of 23, first paragraph of the amendment and response), examiner notes, as cited in the previous office action, *Green* discloses identifying all public classes included in a first object-oriented software

package and identifying all public entities included in each of the public classes, as recited in claims 1, 14 and 27 of the present invention. Similarly, the remaining arguments pertaining to *Krishna* not disclosing *identifying* all public class that are included in the IDE file or all public entities that are in each of the public classes (page 15 of 23, first paragraph of the amendment and response) and *Krishna* not distinguishing between public, private or protected classes (Page 15 of 23, second paragraph) are not persuasive, in respect to *Green* teaching the limitation.

(2) In regard to the argument that Krishna fails to mention anything about removing references to software that is defined in a second object-oriented software package (page 16 of 23, first paragraph), the examiner disagrees. As previously indicated, *Krishna* discloses (Page 2, Paragraph [0025]), that the library source code (second object-oriented software package) is replaced with stubs, thereby replacing the reference to the second object oriented software with a stub enabling the secondary developer to compile class files without an error, therefore the references are resolved without providing the source code. Thus, the secondary developer can compile and develop source code without infringing on the intellectual property rights of an original library developer (E.g., see Krishna, Page 1, Paragraph [0006]). Therefore, *Krishna* teaches removing references to software that is defined in a second object-oriented software package. Furthermore, replacing a reference to software defined in the second object-oriented software package with a stub, is the same as removing reference to software defined in the second object-oriented software package, as claimed for at least the reasons hereinabove. Additionally, the import list is in regard to

a second external file that may exist via inherency as the Java language allows a single inheritance model. Krishna teaches updating the import list to include stubs not references as alleged by the Applicant.

Furthermore, the return value is not a reference to software defined in a second software object-oriented software package (Page 17 of 23, second Paragraph). The return value is a replacement for a reference, thereby removing the reference to software that is defined in a second software package.

(3) Applicant states..."To the contrary, Krishna again teaches away from removing references to software defined in a second software by specifically teaching, in line 735 of Figure 7B, that the import list is updated to refer to classes that are not in the software package if the method return or if the argument type refers to classes that are not in the software package", (page 17 of 23, third paragraph). Applicant then alleges that Krishna updates the import list to include a reference to software that is outside the package instead of removing the reference to software that is outside the package. In fact, the import list is set with information from the IDE file, which contains stubs or replacements for the references to software defined in a second software, thereby removing references to software that is defined in a second object-oriented software package. Applicant then concludes, ..."Therefore, Krishna does not teach or suggest removing all references to software that is defined in a second object-oriented software package from the public entities included in each of the public classes, which Examiner disagrees. *Krishna* discloses updating the import list with public entities, included in each of the non-private classes, (E.g., see Page 4, Paragraph [0048]),

wherein lines 713-719 of figure 7A show that the import list is updated with replacement stubs for the references, thereby *Krishna* does teach removing all references to software that is defined in a second object-oriented software package from the public entities included in each of the public classes.

(4) In regard to Applicants argument (page 17 of 23, fourth paragraph of the amendment and response), that Krishna does not teach generating an equivalent public class for each of the identified public classes, the equivalent public class including equivalent public entities that include no references to the software defined in the second object-oriented software package, the Examiner disagrees. *Krishna* discloses "The Java Card converter resolves package 170 along with the symbol references in the export files and generates the secondary developer's applet CAP file, which is then processed by loader 145B to generate loadable applet 190", wherein the symbol references are resolved by being replaced with the stubs which are the equivalent of the public classes in the perspective of the compiler and if all the public classes were not replaced with stubs then the compiler would result in an error and Krishna's disclosure would not be functional, therefore, it is inherent to Krishna's teaching that all the public classes and entities referenced in a source program are replaced as illustrated. Thus, Krishna indeed teaches generating an equivalent public class for each of the identified public classes, the equivalent public class including equivalent public entities that include no references to the software defined in the second object-oriented software package.

(5) In regard to Applicants' argument (page 18 of 23, second paragraph of the amendment and response), examiner disagrees for at least the reasons hereinabove. Furthermore, Applicant alleges that *Krishna* teaches including an import list that includes references to classes that are not in the same package, which examiner disagrees. Examiner respectfully contends that the import list contains stubs equivalent to the classes in the external package, not references to the classes as stated by Applicant. Thus, the examiner maintains the rejection in respect to the instant argument.

(6) In regard to Applicants' argument that *Green* does not teach or suggest removing all references to software that is defined in a second object-oriented software package (Page 18 of 23, last paragraph), it is noted that *Green* is not relied upon in the previous office action to teach removing all references to software that is defined in a second object-oriented software package.

(7) In response to applicant's argument that there is no suggestion to combine the references (Page 19 of 23, second paragraph), the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, *Green* suggests "...to use the reflection API if you are writing developmental tools..." (Page 1, Paragraph [0001]), and *Krishna's* invention is a method

or tool to facilitate development. Therefore, it would have been obvious to combine *Krishna* and *Green*. Moreover, the remaining arguments of Page 19 of 23, second paragraph, are not persuasive for at least the reasons addressed above.

(8) In view of the above, Examiner maintains the rejections over *Krishna* in view of Green as originally submitted in the previous office action and further clarified hereinabove. Therefore, the rejection of independent claims 1, 14 and 27 and dependent claims 2, 4-7, 9-13, 15, 17-20, 22-26, 28, 30-33 and 35-39 are maintained.

(9) In regard to Applicants' argument that *Evans* does not teach in response to a determination that each of said entities includes a native attribute, removing said native attribute from each of said entities, and generating equivalent entities that include no native attributes (Page 21 of 23, second paragraph), which Examiner disagrees. Applicant states that *Evans* merely teaches editing existing native code that corresponds to an existing method, which Examiner also disagrees. While *Evans* does teach editing existing native code, he also teaches substituting the existing native code. As another example of *Evans* disclosure, Figure 7 & Column 11, lines 13-25 illustrate each class from 1 – N, comprising a stub in place of a native code, wherein the stubs may be replaced by native code, it is also inherent that the stubs may replace native code, thereby removing said native attribute from each of said entities and generating equivalent entities that include no native code. Thus, examiner maintains the rejections over *Krishna in view of Green in further view of Evans* with respect to dependent claims 3, 16 and 29.

(10) The rejections of claims 8, 21 and 34 are maintained. In response to applicant's argument that there is no suggestion to combine the references (Page 22 of 23, second paragraph), the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, *Green* suggests "...to use the reflection API if you are writing developmental tools..." (Page 1, Paragraph [0001]), and *Krishna's* invention is a method or tool to facilitate development. Therefore, it would have been obvious to combine *Krishna* and *Green*. Moreover, the remaining arguments of Page 22 of 23, second paragraph, are not persuasive for at least the reasons addressed above. Thus, the rejections of claims 8, 21 and 34 are maintained.

Claim Rejections

5. Claims 1-39, are pending claims, stand finally rejected in light of the additional clarifications provided and/or addressed at item 2 above, Prior Art's Arguments – Rejections, as claims 1, 2, 4-7, 9-15, 17-20, 22-28, 30-33 and 35-39 are unpatentable over *Krishna in view of Green*. Claims 3, 16 and 29 are unpatentable over *Krishna in view of Green and further in view of Evans*. Claims 8, 21 and 34 are unpatentable over

Krishna in view of Green and further in view of obviousness. The claim rejections from the previous office action of February 24th, 2005 are included corresponding to the pending claims.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims **1, 2, 4-7, 9-15, 17-20, 22-28, 30-33** and **35-39** are rejected under 35 U.S.C. 103(a) as being unpatentable over Krishna et al., US 2003/0051233 A1 (hereinafter **Krishna**) in view of Dale Green, "Trail: The Reflection API", The Java Tutorial. Posted November 27th, 1999. Retrieved from <http://Green.com/docs/books/tutorial/reflect/> (hereinafter **Green**).

3. In regard to claim **1**, **Krishna** discloses:

- "*A method in a data processing system for generating a generic compilation interface from a first object-oriented software package, said method comprising the steps of...*" (E.g., see Figure 7A & Page 2, Paragraph [0025]), wherein the library stubs exclude the source code

executable statements, but include (generic), declarations and interfaces so that the secondary developer can compile class files for converting to CAP files, etc (interface).

- "...*removing all references to software that is defined in a second software package from said public entities included in each of said public classes...*" (E.g., see Figure 7B & Page 2, Paragraph [0025]), wherein the library stubs exclude (remove), the source code executable statements (software defined in a second package), but include declarations and interfaces so that the secondary developer can compile class files, wherein Figure 7B, steps 721- 734, teach replacing a reference returned with the appropriate return type value.
- "...*generating an equivalent public class for each of said identified public classes, said equivalent public class including equivalent public entities that include no references to said software defined in said second package...*" (E.g., see Figure 7A & 7B, steps 737-747 & Page 3, Paragraph [0036]), wherein equivalent public class for each of said identified public class are generated, wherein "...only non-private (public) method signatures and field signatures are needed for off-card compiling and conversion...is sufficient for synthesizing the library stubs 220 (Figure 3)".
- "...*compiling each of said equivalent public classes; and generating a compilation interface for said first package including each of said*

compiled equivalent public classes." (E.g., see Figure 7A & 7B & Page 3, Paragraph [0048]), wherein the pseudo code teaches generating (compiling) an equivalent JAR file (Step 747).

But **Krishna** does not expressly disclose "*...identifying all public classes included in said first software package ...*" or "*...for each of said public classes, identifying all public entities included in each of said public classes ...*". However, **Green** discloses:

- "*...identifying all public classes included in said first software package ...*" (E.g., see "Discovering Class Modifiers", Page 7), wherein all public class modifiers are discovered.
- "*...for each of said public classes, identifying all public entities included in each of said public classes ...*" (E.g., see "Trail: The Reflection API", Page 1, Paragraph 1), wherein all public class modifiers are discovered along with their fields, methods and variables (entities).

Krishna and **Green** are analogous art because they are both concerned with the same field of endeavor, namely, using the JAVA language to examine, manipulate and work with classes. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine a public class modifiers and their attributes with **Krishna's** JAVA program for interpreting, interfacing and compiling. The motivation to do so, is suggested by **Krishna,** "...only non-private method signatures and field signatures are needed... for compiling...", (Page 3, Paragraph [0035]). Furthermore, **Green** suggests "...to use the reflection API if you are writing development tools..." (Page 1, Paragraph 1).

4. In regard to claim **2**, the rejections of base claim **1** are incorporated. Furthermore, **Green** discloses:

- "...*identifying all entities included in each of said public classes that include a public modifier.*" (E.g., see "Trail: The Reflection API", Page 1), wherein all public class modifiers are discovered along with their fields, methods, superclasses and variables (entities).

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine a public class modifiers and their entities with **Krishna's** JAVA program for interpreting, interfacing and compiling. The motivation to do so, is suggested by **Krishna,** "...only non-private method signatures and field signatures are needed... for compiling...", (Page 3, Paragraph [0035]). Furthermore, **Green** suggests "...to use the reflection API if you are writing development tools..." (E.g., see "Trail: The Reflection API", Page 1).

5. In regard to claim **4**, the rejections of base claim **1** are incorporated. Furthermore, **Green** discloses:

- "...*identifying all public methods included in each of said public classes.*" (E.g., see "Obtaining Method Information", Page 15), wherein one can "...uncover a method's name, return type, parameter types, set of modifiers, and set of throwable exceptions."

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine public methods modifiers and their entities

with **Krishna's** JAVA program for interpreting, interfacing and compiling. The motivation to do so, is suggested by **Krishna,** "...only non-private method signatures and field signatures are needed... for compiling...", (Page 3, Paragraph [0035]). Furthermore, **Green** suggests "...to use the reflection API if you are writing development tools..." (E.g., see "Trail: The Reflection API", Page 1, Paragraph 1).

6. In regard to claim **5**, the rejections of base claim **1** are incorporated. Furthermore, **Green** discloses:

- "...*public parameters included in each of said public classes.*" (E.g., see "Obtaining Method Information", Page 15, Paragraph 2), wherein "... the following sample program prints the name, return type, and parameter types of every public method in the Polygon class", wherein the public parameters are included.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine public parameters with **Krishna's** JAVA program for interpreting, interfacing and compiling. The motivation to do so, is suggested by **Krishna,** "...only non-private method signatures and field signatures are needed... for compiling...", (Page 3, Paragraph [0035]). Furthermore, **Green** suggests "...to use the reflection API if you are writing development tools..." (E.g., see "Trail: The Reflection API", Page 1, Paragraph 1).

7. In regard to claim **6**, the rejections of base claim **1** are incorporated. Furthermore, **Green** discloses:

- "...*public fields included in each of said public classes.*" (E.g., see "Trail: The Reflection API", Page 1), wherein all public class modifiers are discovered along with their fields, methods and variables (entities).

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine a public class modifiers and their fields with **Krishna's** JAVA program for interpreting, interfacing and compiling. The motivation to do so, is suggested by **Krishna,** "...only non-private method signatures and field signatures are needed... for compiling...", (Page 3, Paragraph [0035]). Furthermore, **Green** suggests "...get information about a class's ...fields..." (Page 1, Paragraph 1).

8. In regard to claim **7**, the rejections of base claim **1** are incorporated. Furthermore, **Krishna** discloses:

- "...*identifying all public classes included in a Java Archive file.*" (E.g., see Figure 7A, step 704), wherein an IDE file from which to synthesize classes/jar file is disclosed.

9. In regard to claim **9**, the rejections of base claim **1** are incorporated. Furthermore, **Green** discloses:

- "...*identifying all public entities included in each of said public classes utilizing Java Reflection.*" (E.g., see "Trail: The Reflection API", Page 1), wherein all public class modifiers are discovered along with their fields, methods and variables (entities) using the Java Reflection API.

Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine Java Reflection with **Krishna's** JAVA

program for interpreting, interfacing and compiling. The motivation to do so, is suggested by **Krishna,** "...only non-private method signatures and field signatures are needed... for compiling...", (Page 3, Paragraph [0035]). Furthermore, **Green** suggests "...to use the reflection API if you are writing development tools..." (E.g., see "Trail: The Reflection API", Page 1, Paragraph 1).

10. In regard to claim **10**, the rejections of base claim **1** are incorporated. Furthermore, **Krishna** discloses:

- "...*generating a separate java file for each of said identified public classes.*" (E.g., see Figure 7B, step 738), wherein a separate .java file is generated for each of said identified public classes.

11. In regard to claim **11**, the rejections of base claim **10** are incorporated. Furthermore, **Krishna** discloses:

- "...*compiling each said java file.*" (E.g., see Figure 7B, step 744), wherein a separate .java file is compiled.

12. In regard to claim **12**, the rejections of base claim **11** are incorporated. Furthermore, **Krishna** discloses:

- "...*generating a compilation Java Archive file; and storing each said compiled .java file in said compilation Java Archive file.*" (E.g., see Figure 7B, step 747), wherein the class files are placed in a Java Archive file.

13. In regard to claim **13**, the rejections of base claim **1** are incorporated. Furthermore, **Krishna** discloses:

- "...*utilizing said compilation interface within an Integrated Development Environment.*" (E.g., see Figure 1 and Paragraph [0005]), wherein a standard Java development environment is disclosed.

14. As per claims **14, 15, 17-20** and **22-26**, this is a system version of the claimed method discussed above, in claims **1,2, 4-7** and **9-12**, wherein all claimed limitations have also been addressed and/or cited as set forth above. For example, see **Krishna** (Figure 8), wherein a system employing the above methods is disclosed.

15. As per claims **27, 28, 30-33** and **35-39**, this is a product version of the claimed method discussed above, in claims **1,2, 4-7** and **9-12**, wherein all claimed limitations have also been addressed and/or cited as set forth above. For example, see **Krishna** (Figure 1 & Paragraph [0053]), wherein a product version of the above methods is disclosed.

16. Claims **3, 16** and **29** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishna** in view of **Green** and further in view of Evans et al., US 6,836,884 B1 (hereinafter **Evans**).

17. In regard to claim **3**, the rejections of base claim **1** are incorporated. But the combined teaching of **Krishna** and **Green** do not expressly disclose, "...*determining whether each of said entities includes a native attribute; in response to a determination that each of said entities includes a native attribute, removing said native attribute from each of said entities; and generating equivalent entities that include no native attributes.*" However, **Evans** discloses:

- "...*determining whether each of said entities includes a native attribute; in response to a determination that each of said entities includes a native attribute, removing said native attribute from each of said entities; and generating equivalent entities that include no native attributes.*" (E.g., see Figure 3 & Column 12, line 63 – Column 13, line 11), wherein native code may be edited from one form to a more general form.

Evans and the combined teaching of **Krishna** and **Green** are analogous art because they are both concerned with the same field of endeavor, namely, compiling software code. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine replacing native code with the combined teachings method. The motivation to do so, is suggested by **Evans,** "...the user may replace an existing method created in a first source language with a new method created in a second source language.", (E.g., see Column 2, line 65 – Column 3, line 1).

18. As per claim **16**, this is a system version of the claimed method discussed above, in claim **3**, wherein all claimed limitations have also been addressed and/or cited as set forth above. For example, see **Krishna** (Figure 8), wherein a system employing the above method is disclosed.

19. As per claim **29**, this is a product version of the claimed method discussed above, in claim **3**, wherein all claimed limitations have also been addressed and/or cited

as set forth above. For example, see **Krishna** (Figure 1 & Paragraph [0053]), wherein a product version of the above method is disclosed.

20. Claims **8, 21** and **34** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishna** in view of **Green** and further in view of obviousness.

21. In regard to claim **8**, the rejections of base claim **1** are incorporated. But the combined teaching of **Krishna** and **Green** do not expressly disclose, "*...utilizing a java.util.jar utility.*" However, it would have been obvious, to one of ordinary skill in the art to utilize a java.util.jar utility to identify all public classes included in said first package. The motivation to do so is provided by **Green** (E.g., see "Examining Interfaces", Page 1), wherein importing java.util.* is shown, wherein the java.util.jar utility would be included. Furthermore, the java.util.jar is another known method to extract class information and thus it would have been obvious to one of ordinary skill in the art to use a utility already imported to achieve a goal already stated (identify all public classes). Therefore, at the time the invention was made it would have been obvious to identify all public classes included in said first package utilizing a java.util.jar utility with the combined teaching of **Krishna** and **Green**.

22. As per claim **21**, this is a system version of the claimed method discussed above, in claim **8**, wherein all claimed limitations have also been addressed and/or cited as set forth above. For example, see **Krishna** (Figure 8), wherein a system employing the above method is disclosed.

23. As per claim **34**, this is a product version of the claimed method discussed above, in claim **8**, wherein all claimed limitations have also been addressed and/or cited as set forth above. For example, see **Krishna** (Figure 1 & Paragraph [0053]), wherein a product version of the above methods is disclosed.

Conclusion

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to John J Romano whose telephone number is (571) 272-3872. The examiner can normally be reached on 8-5:30, M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

JJR

TUAN DAM
SUPERVISORY PATENT EXAMINER